|  |  |
| --- | --- |
| Project for Database Design—Phase III | DEC 4, 2017 |
|  |  |

Project for Database Design

**Phase IV. DOCUMENTATION**

Anand K Rajagopalan Kartikey Gupta

akr170430@utdallas.edu kxg173430@utdallas.edu

(November 4, 2017)

**0. Pre-Illumination**

# In this project report we will follow the requirement of Phase IV directly. In Section 1 we gave problem description copied from Web site; in Section 2 we answered 3 questions listed in the project and justified our solution; in Section 3 we exhibited EER diagram with all assumptions; in Section 4 we showed our relational schema after normalization; in Section 5 we gave all requested SQL statements for both views and queries; and in Section 6 we gave dependency diagram induced from relational schemas. Finally, a short summary is given at the end of this report.

**1. Problem Description**

Design, develop, and test a database for ABC hotel. The project consists of four parts: conceptual database design (Phase I), logical database design (Phase II), Oracle relational database implementation (Phase III), and final report &demo (Phase IV).

1. The hotel has a group of employees. Each employee has a unique ID number (9-digit number), name, age, address (street number, street name, city, state, zip code), and salary rate. The employees are categorized as discussed below.
2. Employees are categorized based on their job functions: management, reception, dining, housekeeping and concierge, there are also other types (tech support, accountants, etc.) of employees who do not fall into one of the above main types.
3. For each management team member, the system records his/her title (general manager, customer relationship manager, revenue executive, event manager, etc.).
4. For each receptionist, the system tracks a list of languages he/she can speak and read.
5. For each dining staff, the system keeps his/her shift (morning, afternoon, evening, night). Dining staff can be further categorized based on the type of dining they serve: lounge/bar, fine dining restaurant, buffet, catering, etc.

6. For housekeeping staff and concierge, the system tracks their years of experience (integer between 0 to 50). For tech support and accountants, the system tracks the licenses they have obtained.

1. The hotel has two main types of clients: individuals and organizations.
2. The system tracks the information of each individual customer: unique ID (6-digit number), name, sex, phone (may be multiple phone numbers, in the format of ‘(xxx)xxx-xxxx’), and date of birth (both in the format of ‘MM/DD/YYYY’).

9. An individual customer may or may not be a member of the hotel’s rewards program. Information about the membership is recorded for a customer if he/she has one. A

customer may have multiple membership numbers.

10. The hotel tracks the information of hotel rooms. Each room has a unique room number (4-digit number), bed type (twin, king, double queen, etc.), room type (standard, premium, suite, etc.), and per-night price (in USD).

11. Housekeepers clean hotel rooms, the date, time and the housekeeper’s ID are recorded every time a room is cleaned.

12. Receptionists help individual customers check in to their hotel rooms. An individual customer can be helped by different receptionists check in to different hotel rooms during multiple stays at the hotel. Each time a customer is checked in, the check-in date (‘MM/DD/YYYY’), time (‘HH-MM-SS’), key type (card key or digital key), lounge access (Yes/No), and length of the stay are recorded.

13. An individual customer needs to pay his/her bill at check out. Date issued, check-in and checkout date, bill amount in USD are recorded for each bill. A customer may make multiple payments to pay off one bill. The date, time and the payment amount in USD are recorded for each payment.

14. An organization client is uniquely identified by an ID (6-digit number), and may or may not have a direct bill account with the hotel. The account number is recorded if a client has one. A client may have multiple direct account numbers, but the account number itself cannot uniquely identify the account.

15. Organization clients may hold events at the hotel. An organization can hold multiple events, and an event can be held by multiple organizations. For each event, a deposit must be paid to the hotel, and the amount in USD is recorded.

16. The hotel assembles event staff to help serve the events. Four main types of event staff come from management, catering, tech support and accountants. Every event staff member is equipped with an on-call speaker and the on-call number (4-digit number) is recorded in the system. An event staff member can serve multiple events. Each event can be served by a group of event staff, and is uniquely identified by an event ID (4-digit number). The date and time of an event are also recorded. An event manager from the management team is assigned for each event; the manager’s ID (9- digit number) is recorded.

17. An accountant prepares a bill for each event. Each bill has a unique ID (6-digit number), date issued (‘MM/DD/YYYY’) and total amount in USD. An organization may have multiple bills to pay, and each bill may be paid by multiple organizations. An organization can make multiple payments to a bill. Each time an organization makes a payment, the system records the type of payment (cash, check, credit cards, etc.), amount in USD, date and time of the payment.

**2. Three Questions**

**2**.**1**)Can you think more rules (other than the one explicitly described above) that are likely to be used in the system?

1. One individual client can be helped by only one receptionist for one check in.
2. There is exactly one bill associated with each check in.
3. A bill may be paid by an individual or an organization.
4. An event manager can manage multiple events.
5. An event has to be managed by exactly one event manager

**2.2. Is the ability to model super-class/subclass relationships likely to be important in such environment? Why or why not?**

Yes, we believe that the ability to model super-class/subclass relationships is very important for such an environment.With the help of superclass/ subclass relationships we can model complex concepts like attribute inheritance in such an environment.

Here, we have EMPLOYEE categorized based on their job functions: MANAGEMENT, RECEPTION, DINING, HOUSEKEEPING and CONCEIRGE. In such a case having a superclass/ subclass relationship is very important and helps us breakdown to simpler entities.

**2.3 Justify using a Relational DBMS (eg. Oracle) for this project.**

Relational DBMS like oracle provide many advantages such as:

* Reducing the amount of time spent managing data.
* Ability to analyse data in various ways.
* Promotes a disciplined approach to data management.
* Improves the quality and consisitency of information.
* Provides security
* Easy implementation i.e. creation of views, tables, insertion, updation, deletion of data.

Due to these reasons using Relational DBMS like oracle is justified for this project

**3. EER diagram with all assumptions**

## EER Model Assumptions:

1. An EMPLOYEE can have just one address.
2. There can be only two types of licensed employees, TECH\_SUPPORT and ACCOUNTANT.
3. A licensed employee can obtain multiple licenses.
4. Dining staff can be any of the types (i.e. they can overlap. A dining staff that is catering can be part of the other staff members as well).
5. A housekeeper must clean a room and a room must be cleaned by a housekeeper (total participation in both sides).
6. One or more housekeepers can clean one room. One or more rooms can be cleaned by the same housekeeper (Many to Many).
7. Housekeepers’ id is the same as their employee id.
8. An individual client must check into a room (total participation). A room may or may not be checked into by an individual (Partial participation).
9. One or more individual clients can check into a room. One or more rooms can be checked into by an individual client during different visits.
10. One individual client can be helped by only one receptionist for one check in.
11. There is exactly one bill associated with each check in.
12. A bill may be paid by an individual or an organization.
13. An event manager can manage multiple events.
14. An event has to be managed by exactly one event manager.
15. Exactly one accountant prepares exactly one bill for an event.
16. Bill accounts of organizations contain bill for the organization.
17. Each payment is associated with a bill.

## Notation used(EER DIAGRAM AT THE LAST PAGE):

We have used arrows instead of the subset symbol to specify the class/subclass relationship.

**4. Relational Schema in Third Normal Form**

**4.1 Relational Schema**

EMPLOYEE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Emp\_ID | Name | Age | Salary\_rate | Street\_no | Street\_name | Zip\_code |

ZIPCODE\_CITY

|  |  |
| --- | --- |
| Zip\_code | City |

CITY\_STATE

|  |  |
| --- | --- |
| City | State |

HOUSE\_KEEPING

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

CONCIERGE

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

RECEPTION

|  |
| --- |
| Emp\_ID |

LICENSED\_EMP

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Licensed\_emp\_type |

MANAGEMENT

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Title |

DINING

|  |  |
| --- | --- |
| Emp\_ID | Shift\_type |

LANGUAGES

|  |  |
| --- | --- |
| Emp\_ID | License\_obtained |

LICENSES\_OBTAINED

|  |  |
| --- | --- |
| Emp\_ID | Language |

BUFFET

|  |
| --- |
| Emp\_ID |

FINE\_DINING

|  |
| --- |
| Emp\_ID |

LOUNGE/BAR

|  |
| --- |
| Emp\_ID |

EVENT\_STAFF

|  |  |
| --- | --- |
| Event\_staff\_ID | On-Call\_no |

CATERING

|  |  |
| --- | --- |
| Emp\_ID | Event\_staff\_ID |

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

SERVES

|  |  |
| --- | --- |
| Event\_staff\_ID | Event\_ID |

ROOM

|  |  |  |  |
| --- | --- | --- | --- |
| Room\_no | Price\_per\_night | Room\_type | Bed\_type |

CLEANS

|  |  |  |  |
| --- | --- | --- | --- |
| Housekeeping\_ID | Room\_no | Date | Time |

CHECKIN

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Room\_no | Checkin\_date | Bill\_ID | Receptionist\_ID | Individual\_ID | Check\_out\_date | Lounge\_access | Key\_type | Time |

CLIENT

|  |
| --- |
| Client\_ID |

INDIVIDUAL

|  |  |  |  |
| --- | --- | --- | --- |
| Client\_ID | D\_O\_B | Sex | Name |

PHONE\_NO

|  |  |
| --- | --- |
| Client\_ID | Phone\_no |

MEMBERSHIP\_NO

|  |  |
| --- | --- |
| Client\_ID | MEMBERSHIP\_NO |

ORGANIZATION

|  |
| --- |
| Client\_ID |

BILL

|  |  |  |
| --- | --- | --- |
| Bill\_id | Bill\_amount | Date\_issued |

PAYMENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bill\_ID | Payment\_ID | Client\_ID | date | Time | Amount | Type |

PREPARES

Bill\_ID Accountant\_ID Event\_ID

ACCOUNT

|  |  |  |
| --- | --- | --- |
| Client\_ID | Account\_No | Account\_type |

CONTAINS

Client\_ID Account\_no Bill\_ID

HOLDS

Event\_ID Organizational\_ID Deposit

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

HOLDS

Event\_ID Organizational\_ID Deposit

**4.2 Format for Every Relation**

Table 3 gives data type and format for each attribute in each relational schema.

Table 3. Format for Each Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |  |
| EMPLOYEE | Emp\_id | Integer |  |
|  | Name | String <= 20 chars |  |
|  | Age | Integer [0, 200] |  |
|  | Salary\_rate | Float |  |
|  | Street\_No | Integer |  |
|  | Street\_Name | String <= 20 chars |  |
|  | City | String <= 20 chars |  |
|  | State | String <= 20 chars |  |
|  | Zip\_code | String <= 6 chars |  |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| HOUSE\_KEEPING | Emp\_id | Integer |
|  | Yrs\_experience | Integer [0, 50] |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CONCIERGE | Emp\_id | Integer |
|  | Yrs\_experience | Integer [0, 50] |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| LICENSED\_EMP | Emp\_id | Integer |
|  | Event\_staff\_id | Integer |
|  | Licensed\_employee\_type | String {‘ACCOUNTANT’, ‘TECH\_SUPPORT’} |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| RECEPTION | Emp\_id | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| MANAGEMENT | Emp\_id | Integer |
|  | Event\_staff\_id | Integer |
|  | Title | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| DINING | Emp\_id | Integer |
|  | Shift\_type | Char {m, a, e, n} |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| LICENSES\_OBTAINED | Emp\_id | Integer |
|  | License\_obtained | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| LANGUAGES | Emp\_id | Integer |
|  | Language | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| LOUNGE\_BAR | Emp\_id | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| FINE\_DINING | Emp\_id | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| BUFFET | Emp\_id | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CATERING | Emp\_id | Integer |
|  | Event\_staff\_id | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| EVENT\_STAFF | Event\_staff\_id | Integer |
|  | On-call\_no | Integer 4 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| EVENT | Event\_id | Integer 4 digits |
|  | Event\_staff\_id | Integer |
| Manager\_id | Integer |
| Time | Time |
| Date | Date |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| SERVES | Event\_staff\_id | Integer |
|  | Event\_id | Integer 4 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| ROOM | Room\_no | Integer 4 digits |
|  | Price\_per\_night | Float |
| Room\_type | String <= 20 chars |
| Bed\_type | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CLEANS | Housekeeping\_id | Integer |
|  | Room\_no | Integer 4 digits |
| Time | Time |
| Date | Date |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CHECKIN | Room\_no | Integer 4 digits |
|  | Checkin\_date | Date |
| Bill\_id | Integer 6 digits |
| Receptionist\_id | Integer |
| Individual\_id | Integer 6 digits |
| Checkout\_date | Date |
| Lounge\_access | Boolean |
| Key\_type | String {‘card’, ‘digital'} |
| Time | Time |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CLIENT | Client\_id | Integer 6 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| INDIVIDUAL | Client\_id | Integer 6 digits |
|  | d\_o\_b | Date |
| Sex | String <= 20 chars |
| Name | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| PHONE\_NO | Client\_id | Integer 6 digits |
|  | Phone\_no | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| MEMBERSHIP\_NO | Client\_id | Integer 6 digits |
|  | Membership\_no | Integer |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| ORGANIZATION | Client\_id | Integer 6 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| BILL | Bill\_id | Integer 6 digits |
|  | Bill\_amount | Float |
| Date\_issued | Date |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| PAYMENT | Bill\_id | Integer 6 digits |
|  | Payment\_id | Integer |
| Client\_id | Integer 6 digits |
| Time | Time |
| Date | Date |
| Amount | Float |
| Type | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| PREPARES | Bill\_id | Integer 6 digits |
|  | Accountant\_id | Integer |
| Event\_id | Integer 4 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| ACCOUNT | Client\_id | Integer 6 digits |
|  | Account\_no | Integer |
| Account\_type | String <= 20 chars |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| CONTAINS | Client\_id | Integer 6 digits |
|  | Account\_no | Integer |
| Bill\_id | Integer 6 digits |

|  |  |  |
| --- | --- | --- |
| **Relation Names** | **Attributes** | **Date Type** |
| HOLDS | Event\_id | Integer 4 digits |
|  | Organization\_id | Integer |
| Deposit | Float |

**5. All Requested SQL Statements**

**5.1 Creation of Database with SQL Statements**

**5.1.1 Table Creation**

**3.1.1 Table Creation**

Using SQL statement, we created tables as follows:

CITY\_STATE TABLE:

create table CITY\_STATE

(city varchar(30),

state\_ varchar(30),

primary key(city));

ZIPCODE\_CITY TABLE:

create table ZIPCODE\_CITY

(zip\_code INT,

city varchar(30),

primary key(zip\_code),

foreign key(city) references CITY\_STATE(city));

EMPLOYEE TABLE:

create table EMPLOYEE

(Emp\_id INT,

Name varchar(20),

Age INT,

Salary\_rate Float,

Street\_no INT,

Street\_name varchar(30),

zip\_code INT,

primary key(Emp\_id),

foreign key(zip\_code) references ZIPCODE\_CITY(zip\_code));

HOUSE\_KEEPING TABLE:

create table HOUSE\_KEEPING

(Emp\_id INT,

yrs\_experience INT check(yrs\_experience >=0 and yrs\_experience<=50),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

CONCIERGE TABLE:

create table CONCIERGE

(Emp\_id INT,

yrs\_experience INT check(yrs\_experience >=0 and yrs\_experience<=50),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

LICENSED\_EMP TABLE:

create table LICENSED\_EMP

(Emp\_id INT,

Event\_staff\_id INT,

Licensed\_Employee\_type varchar(30) check( Licensed\_Employee\_type in ('ACCOUNTANT', 'TECH\_SUPPORT', 'Accountant', 'Tech\_support')),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

EVENT\_STAFF TABLE:

create table EVENT\_STAFF

(Event\_staff\_id INT,

oncall\_no INT,

primary key(Event\_staff\_id));

RECEPTION TABLE:

create table RECEPTION

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

MANAGEMENT TABLE:

create table MANAGEMENT

(Emp\_id INT,

Event\_staff\_id INT,

Title varchar(20),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

EVENT TABLE:

create table EVENT

(Event\_ID INT,

Manager\_ID INT,

Time\_ TIMESTAMP,

Date\_ DATE default(Sysdate),

primary key(Event\_ID),

foreign key (Manager\_ID) references MANAGEMENT(Emp\_id));

DINING TABLE:

create table DINING

(Emp\_id INT,

Shift\_type char(1) check(Shift\_type in ('m', 'a', 'e', 'n')),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

LOUNGE\_BAR TABLE:

create table LOUNGE\_BAR

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

FINE\_DINING TABLE:

create table FINE\_DINING

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

BUFFET TABLE:

create table BUFFET

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

CATERING TABLE:

create table CATERING

(Emp\_id INT,

Event\_staff\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

SERVES TABLE:

create table SERVES

(Event\_staff\_id INT,

Event\_ID INT,

primary key(Event\_staff\_id,Event\_ID),

foreign key(Event\_staff\_id) references EVENT\_STAFF,

foreign key(Event\_ID) references EVENT(Event\_ID));

LANGUAGES TABLE:

create table LANGUAGES

(Emp\_id INT,

language varchar(20),

primary key(Emp\_id, language),

foreign key(Emp\_id) references RECEPTION(Emp\_ID));

LICENSES\_OBTAINED TABLE:

create table LICENSES\_OBTAINED

(Emp\_id INT,

License\_obtained varchar(20),

primary key (Emp\_id, License\_obtained),

foreign key(Emp\_id) references LICENSED\_EMP(Emp\_id));

ROOM TABLE:

create table ROOM

(Room\_no INT,

price\_per\_night float,

Room\_type varchar(20),

Bed\_type varchar(20),

primary key(Room\_no));

CLEANS TABLE:

create table CLEANS

(Housekeeping\_id INT,

Room\_no INT,

date\_ DATE default(sysdate),

time\_ Timestamp,

Primary key(Housekeeping\_id, Room\_no),

foreign key(Housekeeping\_id) references HOUSE\_KEEPING(Emp\_id),

foreign key(Room\_no) references ROOM(Room\_no));

CLIENT TABLE:

create table CLIENT

(client\_id INT,

primary key(client\_id));

BILL TABLE:

create table BILL

(Bill\_id INT,

Bill\_amount float,

Date\_issued Date,

primary key(Bill\_id));

PREPARES TABLE:

create table PREPARES

(Bill\_id INT,

Account\_id INT,

Event\_id INT,

primary key(Bill\_id),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(Account\_id) references LICENSED\_EMP(Emp\_id),

foreign key(Event\_id) references EVENT(Event\_id));

ORGANIZATION TABLE:

create table ORGANIZATION

(client\_id INT,

primary key(client\_id),

foreign key(client\_id) references CLIENT(client\_id));

ACCOUNT TABLE:

create table ACCOUNT

(client\_id INT,

Account\_no INT,

Account\_type varchar(20),

primary key(client\_id, Account\_no),

foreign key(client\_id) references ORGANIZATION(client\_id));

HOLDS TABLE:

create table HOLDS

(Event\_id INT,

Organization\_id INT,

Deposit float,

primary key(Event\_id, Organization\_id),

Foreign key(Organization\_id) references ORGANIZATION(client\_id),

foreign key(Event\_id) references EVENT(Event\_id));

CONTAINS TABLE:

create table CONTAINS

(client\_id INT,

Account\_no INT,

Bill\_id INT,

primary key(client\_id, Account\_no, Bill\_id),

foreign key(client\_id, Account\_no) references ACCOUNT(client\_id, Account\_no),

foreign key(Bill\_id) references BILL(Bill\_id));

PAYMENT TABLE:

create table PAYMENT

(bill\_id INT,

Payment\_id INT,

client\_id INT,

date\_ Date,

time\_ Timestamp,

amount float,

type varchar(20),

primary key(Bill\_id, Payment\_id),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(client\_id) references CLIENT(client\_id));

INDIVIDUAL TABLE:

create table INDIVIDUAL

(client\_id INT,

D\_O\_B date,

sex varchar(20),

name varchar(20),

primary key(client\_id),

foreign key(client\_id) references CLIENT(client\_id));

CHECKIN TABLE:

create table CHECKIN

(Room\_no INT,

checkin\_date DATE,

Bill\_id INT,

Receptionist\_id INT,

Individual\_id INT,

checkout\_date DATE,

lounge\_access INT,

Key\_type varchar(10) check(Key\_type in('CARD','DIGITAL')),

Time\_ Timestamp,

primary key(Room\_no, checkin\_date),

foreign key(Room\_no) references ROOM(Room\_no),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(Receptionist\_id) references RECEPTION(Emp\_id),

foreign key(Individual\_id) references INDIVIDUAL(client\_id));

PHONE\_NO TABLE:

create table PHONE\_NO

(client\_id INT,

Phone\_no INT,

primary key(client\_id, Phone\_no),

foreign key(client\_id) references CLIENT(client\_id));

MEMBERSHIP\_NO TABLE:

create table MEMBERSHIP\_NO

(client\_id INT,

Membership\_no INT,

primary key(client\_id, Membership\_no),

foreign key(client\_id) references CLIENT(client\_id));

**5.1.2 A Database State**

Put your instance here.

**5.2 Creation of Views (Answer for Question d/Phase III)**

**Available rooms**

CREATE VIEW AVAILABLE\_ROOMS

AS (SELECT R.Room\_no

FROM ROOM R)

MINUS

(SELECT R1.Room\_no

FROM ROOM R1, CHECKIN C

WHERE R1.Room\_no = C.Room\_no AND C.Checkin\_date < SYSDATE AND C.Checkout\_date > SYSDATE);

**Popular event manager**

CREATE VIEW POPULAR\_EVENT\_MANAGER

AS SELECT EMP.\*, E.N\_events

FROM EMPLOYEE EMP,

(SELECT EVE.Manager\_id AS Manager\_id, COUNT(\*) AS N\_events

FROM EVENT EVE

GROUP BY EVE.Manager\_id, EXTRACT(month FROM EVE.DATE\_)) E

WHERE E.Manager\_id = EMP.Emp\_id AND E.N\_events > 10;

**Frequent customers**

CREATE VIEW FREQUENT\_CUSTOMERS

AS SELECT IND.\*, I.N\_checkin

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, COUNT(\*) AS N\_checkin

FROM CHECKIN C

WHERE EXTRACT(year FROM C.TIME\_) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.N\_checkin >= 10;

**Popular rooms**

CREATE VIEW POPULAR\_ROOMS

AS SELECT R.\*, R1.N\_checkin

FROM ROOM R,

(SELECT C.Room\_no AS Room\_no, COUNT(\*) AS N\_checkin

FROM CHECKIN C

WHERE EXTRACT(year FROM C.CHECKIN\_DATE) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Room\_no) R1

WHERE R.Room\_no = R1.Room\_no AND R1.N\_checkin >= 30;

**5.3 Creation of SQL Queries (Answer for Question e/Phase**

**III)**

Now we give out the SQL Queries for all questions listed in **Question e** as follows:

**Retrieve the number of employees who work at the lounge/bar**

SELECT COUNT(\*) AS NO\_OF\_EMPLOYEES FROM LOUNGE\_BAR;

**Retrieve the average salary of the receptionists**

SELECT AVG(E.Salary\_rate) AS AVERAGE\_SALARY

FROM EMPLOYEE E, RECEPTION R

WHERE E.Emp\_id = R.Emp\_id;

**Retrieve the information of individual customers who have been billed more than $1,000 in total this year**

SELECT IND.\*

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, SUM(B.Bill\_amount) AS Total\_bill\_amount

FROM CHECKIN C, BILL B

WHERE C.Bill\_id = B.Bill\_id

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.Total\_bill\_amount > 1000;

**For each individual, retrieve his/her bill amount in ascending order of each check-in date**

SELECT IND.\*, I.Bill\_amount, I.Checkin\_date

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, C.Checkin\_date AS Checkin\_date, B.Bill\_amount AS Bill\_amount

FROM CHECKIN C, BILL B) I

WHERE IND.Client\_id = I.Individual\_id

ORDER BY I.Checkin\_date;

**Retrieve the information of the frequent customers who have stayed for at least 15 nights this year**

SELECT IND.\*

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, SUM(C.Checkout\_date - C.Checkin\_date) AS N\_days\_of\_stay

FROM CHECKIN C

WHERE EXTRACT(year FROM C.Checkin\_date) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.N\_days\_of\_stay >= 15;

**Retrieve the average age of individual customers who were helped by a receptionist who only speaks Spanish**

SELECT AVG((SYSDATE - I.d\_o\_b)/365.242199) AS AVERAGE\_AGE

FROM INDIVIDUAL I

WHERE I.Client\_id IN

(SELECT C.Individual\_id

FROM CHECKIN C,

(SELECT L1.Emp\_id AS Emp\_id

FROM LANGUAGES L1

MINUS

SELECT L2.Emp\_id AS Emp\_id

FROM LANGUAGES L2

WHERE L2.Language != 'Spanish') R

WHERE C.Receptionist\_id = R.Emp\_id);

**Retrieve the information of the organization that organized at least two events and got bills of over $2000 in total**

SELECT ORG1.Client\_id

FROM (SELECT ORG3.Organization\_id AS Client\_id

FROM (SELECT H.Organization\_id, COUNT(\*) AS Count\_events

FROM HOLDS H

GROUP BY H.Organization\_id) ORG3

WHERE ORG3.Count\_events >= 2) ORG1,

(SELECT ORG4.Client\_id AS Client\_id

FROM (SELECT C.Client\_id, SUM(B.Bill\_amount) AS Total\_bill\_amount

FROM CONTAINS C, BILL B

WHERE B.Bill\_id = C.Bill\_id

GROUP BY C.Client\_id) ORG4

WHERE ORG4.Total\_bill\_amount > 2000) ORG2

WHERE ORG1.Client\_id = ORG2.Client\_id;

**Retrieve the highest amount of bill of the events helped by the most popular event manager**

SELECT MAX(B.Bill\_amount) AS MAXIMUM\_BILL\_AMOUNT

FROM EVENT E, BILL B, PREPARES P,

(SELECT PM.\*

FROM (SELECT \*

FROM POPULAR\_EVENT\_MANAGER

ORDER BY N\_events DESC) PM

WHERE rownum = 1) MPM

WHERE MPM.Emp\_id = E.Manager\_id AND P.Event\_id = E.Event\_id

AND P.Bill\_id = B.Bill\_id;

**Retrieve information of the event that each of its organizers pays the highest amount for the event (suppose organizers of the same event pay the bill evenly)**

SELECT E.\*

FROM EVENT E, PREPARES P,

(SELECT O.Bill\_id AS BIll\_id

FROM (SELECT Bill\_id, Client\_id

FROM PAYMENT

GROUP BY (PAYMENT.Bill\_id, PAYMENT.Client\_id)

ORDER BY SUM(PAYMENT.Amount) DESC) O

WHERE rownum = 1) PAY

WHERE E.Event\_id = P.Event\_id AND P.Bill\_id = PAY.Bill\_id;

**Retrieve the date and time the most popular room was last checked in**

SELECT MAX(C.TIME\_) AS Last\_checkin

FROM CHECKIN C,

(SELECT PR.\*

FROM (SELECT \*

FROM POPULAR\_ROOMS

ORDER BY N\_CHECKIN DESC) PR

WHERE rownum = 1) R

WHERE C.Room\_no = R.Room\_no;

**6. Dependency Diagram**

We now draw a dependency diagram for each table from Figure 2 as follows:

EMPLOYEE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Emp\_ID | Name | Age | Salary\_rate | Street\_no | Street\_name | Zip\_code |

ZIPCODE\_CITY

|  |  |
| --- | --- |
| Zip\_code | City |

CITY\_STATE

|  |  |
| --- | --- |
| City | State |

HOUSE\_KEEPING

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

CONCIERGE

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

RECEPTION

|  |
| --- |
| Emp\_ID |

LICENSED\_EMP

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Licensed\_emp\_type |

MANAGEMENT

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Title |

DINING

|  |  |
| --- | --- |
| Emp\_ID | Shift\_type |

LANGUAGES

|  |  |
| --- | --- |
| Emp\_ID | License\_obtained |

LICENSES\_OBTAINED

|  |  |
| --- | --- |
| Emp\_ID | Language |

BUFFET

|  |
| --- |
| Emp\_ID |

FINE\_DINING

|  |
| --- |
| Emp\_ID |

LOUNGE/BAR

|  |
| --- |
| Emp\_ID |

EVENT\_STAFF

|  |  |
| --- | --- |
| Event\_staff\_ID | On-Call\_no |

CATERING

|  |  |
| --- | --- |
| Emp\_ID | Event\_staff\_ID |

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

SERVES

|  |  |
| --- | --- |
| Event\_staff\_ID | Event\_ID |

ROOM

|  |  |  |  |
| --- | --- | --- | --- |
| Room\_no | Price\_per\_night | Room\_type | Bed\_type |

CLEANS

|  |  |  |  |
| --- | --- | --- | --- |
| Housekeeping\_ID | Room\_no | Date | Time |

CHECKIN

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Room\_no | Checkin\_date | Bill\_ID | Receptionist\_ID | Individual\_ID | Check\_out\_date | Lounge\_access | Key\_type | Time |

CLIENT

|  |
| --- |
| Client\_ID |

INDIVIDUAL

|  |  |  |  |
| --- | --- | --- | --- |
| Client\_ID | D\_O\_B | Sex | Name |

PHONE\_NO

|  |  |
| --- | --- |
| Client\_ID | Phone\_no |

MEMBERSHIP\_NO

|  |  |
| --- | --- |
| Client\_ID | MEMBERSHIP\_NO |

ORGANIZATION

|  |
| --- |
| Client\_ID |

BILL

|  |  |  |
| --- | --- | --- |
| Bill\_id | Bill\_amount | Date\_issued |

PAYMENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bill\_ID | Payment\_ID | Client\_ID | date | Time | Amount | Type |

PREPARES

Bill\_ID Accountant\_ID Event\_ID

ACCOUNT

|  |  |  |
| --- | --- | --- |
| Client\_ID | Account\_No | Account\_type |

CONTAINS

Client\_ID Account\_no Bill\_ID

HOLDS

Event\_ID Organizational\_ID Deposit

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

HOLDS

Event\_ID Organizational\_ID Deposit

**7. Conclusion**

In this final report we summarized all the necessary descriptions and solutions for ABC HOTEL database, including process and result of EER diagrams, relational schemas in third normal form, SQL statements to create database, create view and solve corresponding queries, as well as dependency diagram. We also implement the whole database in Oracle and using a database state to test every query. In section 2 we also explained why we use superclass/subclass relationship to build relational schema, why we choose a Relational DBMS to implement our database, and the additional five business rules shown from implementation.